B-FEE-FIX-4-PREFLIGHT — Read-only Audit

Steve Trading Bot — Fee-Struktur-Audit

Datum: 2026-05-08 (UTC)

Master-Tip: `043e3b4`

Bot-PID: 3736 (B-OUTAGE-RESILIENCE-1 aktiv)

Modus: read-only Audit, kein Code, kein Fix.

---

1. Fee-System-Kartographie

1.1 Wo Fees gesetzt werden (live_trade.py)

Funktion	Zeile	Pfad	Fee-Quelle
`_parse_filled`	320	hauptpfad	`_extract_fee_in_quote(order, sym, cost, avg)`
`execute_buy`	531	normal	aus `_parse_filled`
`execute_buy` (balance-diff fallback)	538	wenn filled fehlt	`cost * TRADE_FEE`
`execute_buy` (DCA-add)	567	additiv	bestehender + neuer fee_buy
`execute_sell`	659	normal	aus `_parse_filled`
`execute_sell` (balance-diff fallback)	665	wenn filled fehlt	`gross * TRADE_FEE`
`execute_sell` (fee_buy_total Lookup)	675	für PnL	`pos['fee_buy']` oder `pos['cost'] * TRADE_FEE`
`execute_tier3_buy`	785 / 791	normal / fallback	identisch zu execute_buy
`execute_partial_sell`	analog	normal / fallback	analog

1.2 PaperTrader (`paper_trade.py`)

Funktion	Zeile	Berechnung
`execute_buy`	199	`fee_buy = round(cost * TRADE_FEE, 4)` — immer notional × 0.1%
`execute_sell`	292	`fee_sell = round(gross_proceeds * TRADE_FEE, 4)`
`execute_sell` PnL	296–299	`fee_buy = pos['fee_buy'] or pos['cost'] * TRADE_FEE`

→ PaperTrader nutzt deterministisch notional × TRADE_FEE und ist nicht von API-Daten abhängig. Einzig live-Pfad ist betroffen.

1.3 Aufrufer von `execute_buy`

```
trading/main.py:614, 626, 649, 759, 841, 1007    Legacy-Pipeline
trading/strategies/multi_strategy_runner.py:374   Multi-Strategy
```

Alle Aufrufer routen über die selbe LiveTrader-Instance.

---

2. Aktuelles Verhalten bei leeren Fee-Daten

2.1 `_extract_fee_in_quote` — Branch-Analyse

```
if cost is None or cost <= 0:           # Rule 1
    return 0.0
fee_obj = order.get('fee') or {}
fee_cost_raw = fee_obj.get('cost')
if fee_cost_raw is None:                 # Rule 2
    return float(cost) * TRADE_FEE       # einziger explicit-zero-fallback
try:
    fee_cost = float(fee_cost_raw)       # Rule 7
except (TypeError, ValueError):
    return float(cost) * TRADE_FEE
fee_currency = ...
if not fee_currency:                     # Rule 5
    return float(cost) * TRADE_FEE
if fee_currency == quote_asset:          # Rule 3
    return fee_cost                      # AKZEPTIERT 0.0!
if fee_currency == base_asset:           # Rule 4
    return round(fee_cost * float(avg_price), 8)  # AKZEPTIERT 0.0 * x = 0!
return float(cost) * TRADE_FEE           # Rule 6 (foreign)
```

2.2 Fall-Matrix

Order-fee-Form	fee_cost_raw	Helper-Verhalten	Resultat
`fee = None`	`None` (via `or {}`)	Rule 2 fires	`cost × 0.001` ✓
`fee = {}`	`None`	Rule 2 fires	`cost × 0.001` ✓
`fee = {'cost': None}`	`None`	Rule 2 fires	`cost × 0.001` ✓
`fee = {'cost': 'abc'}`	non-numeric	Rule 7 fires	`cost × 0.001` ✓
`fee = {'cost': 0.0, 'currency': 'XYZ'}`	`0.0`	Rule 7 ok, Rule 3/4	**`0.0` returned** FAIL
`fee = {'cost': 0.5, 'currency': 'USDT'}`	`0.5`	Rule 3	`0.5` ✓
`fee = {'cost': 0.001, 'currency': 'BTC'}`	`0.001`	Rule 4	`0.001 × avg_price` ✓
`fee = {'cost': 0.0, 'currency': 'USDT'}`	`0.0`	Rule 3	**`0.0` returned** FAIL

→ **Bug**: explicit `fee.cost = 0` mit gültiger currency wird als „Trade ohne Fees" akzeptiert. Auf Testnet ist das immer der Fall.

2.3 Real-Beweis aus `fetchMyTrades`

```
APE/USDT (order 30704, fill 170):
  fee = {'currency': 'APE', 'cost': 0.0}
  raw info: commission=0.00000000 commissionAsset=APE isBuyer=True

ALGO/USDT (order 24467, fill 409):
  fee = {'currency': 'ALGO', 'cost': 0.0}
  raw info: commission=0.00000000 commissionAsset=ALGO

LINK/USDT BUY (order 8346):
  fee = {'currency': 'LINK', 'cost': 0.0}

LINK/USDT SELL (order 10720):
  fee = {'currency': 'USDT', 'cost': 0.0}
```

Bestätigt:

- Binance-Testnet folgt der dokumentierten Convention (BUY-fee in base, SELL-fee in quote)

- BUT cost ist immer 0 auf Testnet (testnet erhebt keine echten Fees)
- info.commission auch 0
- Helper sieht 0 als legitim → state.fee_buy = 0

2.4 `fetch_order` vs `fetch_my_trades` — Diskrepanz

Aufruf	order['fee']
`fetch_order(oid)` jetzt (Stunden später)	`null`, `fees=[]`, `trades=[]`
`fetch_my_trades(symbol)`	`{'currency': 'APE', 'cost': 0.0}`

ccxt's `fetch_order`-Aggregation null-out fee wenn alle trades fee=0 haben. `fetch_my_trades` zeigt die per-fill-Realität: explicit zero. Im Moment des Buys hatte ccxt vermutlich `fee = {'cost': 0, 'currency': 'APE'}` → Helper akzeptierte 0.

---

3. Binance- / ccxt-Fee-Quellen

3.1 Order-Response

Feld	Spec	Beobachtet (Testnet)
`order['fee']`	aggregierte Fee	manchmal None (fetch_order) oder `{cost:0, currency:base/quote}` (frisch)
`order['fees']`	Liste pro Fill	leeres Array nach fetch_order
`order['trades']`	per-fill Liste	leeres Array nach fetch_order
`fee.cost`	Numeric	Immer 0 auf Testnet
`fee.currency`	base/quote/BNB	Folgt buy=base / sell=quote convention
`fee.rate`	Optional	Nicht in Testnet-Response gesehen
`filled`	Numeric	OK in Testnet
`cost`	Numeric (notional)	OK in Testnet
`average`	Numeric (avg-fill-price)	OK in Testnet

3.2 Trade-/Fill-Abfrage

Aufruf	Status auf Testnet
`fetch_my_trades(symbol)`	OK funktioniert, gibt per-fill Records
`fetch_my_trades(symbol, since=ts, limit=N)`	unterstützt
`info.commission` (raw)	Numeric-string wie `"0.00000000"`
`info.commissionAsset`	base/quote/BNB
Mehrere Fills pro Order	strukturell unterstützt

3.3 Antworten auf die Detail-Fragen

Frage	Antwort
Liefert Testnet bei Market-Buy zuverlässig fee?	Nein — strukturell ja (fee dict mit currency+cost), aber cost=0 ist nicht informativ.
Liefert Testnet bei Market-Sell zuverlässig fee?	Gleiches Problem — fee.currency=USDT, cost=0
Sind Fees in Base/Quote/Fremdasset möglich?	Ja: BUY default in Base, SELL default in Quote, BNB-discount lässt commissionAsset=BNB werden
Kann commissionAsset BNB sein?	Ja, auf Mainnet wenn BNB-Balance vorhanden + BNB-Fee-Pay aktiviert
Unterschiede Spot Testnet vs Mainnet?	Ja, signifikant: Testnet erhebt keine Fees → cost=0. Mainnet erhebt 0.1% bzw. 0.075% mit BNB-discount
Order-Response vs Trade-Abfrage — was ist zuverlässiger?	Trade-Abfrage (`fetchMyTrades`) ist autoritativ

---

4. Architektur-Empfehlung

4.1 Bewertung der vier Optionen

Option	Pro	Contra	Empfehlung
A: nur `notional × TRADE_FEE`	deterministisch, einfach	ignoriert BNB-discount, ignoriert echte Maker-Fees, ignoriert mehrere Fills	NEIN
B: nur `order['fee']`	exakt was Binance abrechnet	Testnet liefert immer 0; bei `fee=None` auch 0	NEIN
C: `order['fee']` bevorzugen, sonst Fallback	kombiniert Vorteile	aktuell so: aber „explicit zero" wird als gültig akzeptiert (Bug-Quelle)	nur mit zero-detection
D: `fetchMyTrades` + Fallback	autoritativ, BNB-discount-fähig, multi-fill-fähig	extra API-Call, Race-Condition, Retry nötig	Ideal als secondary, nicht alleine

4.2 Empfohlene Zielarchitektur (eigene Ableitung)

Hybrid-Strategie mit Plausibilitäts-Check + multi-source-aggregation:

```
Priorität (von hoch nach niedrig):

1. fetchMyTrades(orderId, symbol) — wenn verfügbar:
   a) Filter trades by orderId
   b) Pro fill: commission + commissionAsset → quote-normalisieren
      - if commissionAsset == quote_asset: use as-is
      - if commissionAsset == base_asset: × fill.price → quote
      - if commissionAsset == BNB: BNB×ticker_BNB/quote (oder Fallback)
      - if commissionAsset unknown: SKIP this fill, use proportional fallback
   c) Aggregate über alle fills
   d) Plausibility-Check: 0 < total_fee < notional × MAX_PLAUSIBLE_RATE (z.B. 0.5%)
   e) Wenn plausibel → return total_fee
   f) Wenn unplausibel ODER total_fee == 0 → goto 2

2. order['fees'] (Liste) — wenn vorhanden + nicht-leer:
   a) Pro fee dict: gleiche Quote-Normalisierung
   b) Aggregate, Plausibility-Check
   c) Wenn 0 → goto 3

3. order['fee'] (Single) — wenn fee.cost > 0:
   a) Quote-Normalisierung
   b) Plausibility-Check
   c) Wenn fee.cost == 0 oder None → goto 4

4. Notional-Fallback: cost × TRADE_FEE (current default 0.1%)
   - Mainnet-aware: TRADE_FEE könnte aus settings je nach VIP-Tier kommen
   - BNB-discount-aware: optional × 0.75 wenn settings.BNB_FEE_PAY=true

5. Logging:
   - Welche Quelle wurde verwendet (myTrades / order.fees / order.fee / fallback)
   - Wenn fallback: warum (zero-fee, unplausible, missing source)
```

4.3 Begründung

1. **`fetchMyTrades` zuerst** — höchste Genauigkeit, BNB-discount-fähig, multi-fill-Support. Race-Condition durch retry mit kurzem backoff (z.B. 2× 0.5s); Fallback bleibt verfügbar.

2. **Plausibilitäts-Check** — verhindert Akzeptanz einer 0-Fee-Antwort wenn echte Fees zu erwarten sind. Konservativ: Mainnet-Spot-Default ist 0.1%, also wenn API 0 sagt aber Notional > 1 USDT → fallback.

3. **Notional-Fallback als letzte Linie** — garantiert immer eine Zahl, deterministisch, dokumentiert.

4. **Multi-Source-Logging** — wenn später ein PnL nicht stimmt, ist klar welcher Pfad verwendet wurde.

4.4 Edge-Cases

Edge-Case	Verhalten
Maker-Order mit Mainnet-Rebate (negative fee)	API-Fee verwenden auch wenn negativ
BNB-Pay-Discount (-25%)	API-Fee in BNB-currency → Konvertierung via ticker oder konservativer Fallback
Multi-Fill mit unterschiedlichen Preisen	nur fetchMyTrades zeigt korrekte per-fill-Aggregation
Testnet (cost=0 strukturell)	Plausibility-Check fängt → fallback `notional × TRADE_FEE`
Mainnet promo (echte 0-Fee promotion)	settings.ALLOW_ZERO_FEE_PROMO wenn Operator weiß was er tut
commissionAsset nicht eindeutig zuordbar	proportional fallback per fill
ccxt-stub-Response (kurz nach Order)	retry myTrades + Fallback

---

5. Validierung der vorgeschlagenen Ziel-Architektur

Die User-skizzierte Architektur (fetchMyTrades primary, order['fees']/['fee'] secondary, Quote-Normalisierung, Aggregation, Fallback) ist **konsistent mit der unabhängigen Analyse**.

User-Punkt	Konkretisierung
1. fetchMyTrades primary	mit kurzem Retry (1–2× 0.5s) wegen Race-Condition
2. order['fees']/['fee'] secondary	mit Plausibility-Check (cost > 0 UND < notional × MAX_RATE)
Quote-Normalisierung	gleiche 8-Rule-Logik wie B-FEE-FIX-1, plus BNB-Sonderfall
Aggregation über Fills	Pflicht für multi-fill-Orders
Fallback notional × TRADE_FEE	als final-line-of-defense

Zusatz-Empfehlung: `fee_source: "my_trades" | "order.fees" | "order.fee" | "fallback_notional"` als Feld in closed_trades, damit später Debugging möglich ist.

---

6. Boundary-Bestätigungen

Punkt	Status	Beleg
Read-only Audit	OK	nur `fetch_order` + `fetch_my_trades` (idempotente reads); keine ccxt-Order-Calls
Keine Mutation	OK	live_portfolio.json hash unverändert seit Audit-Start
Kein Bot-Restart	OK	PID 3736 weiterläuft
Keine Orders	OK	nur Read-Calls
Keine .env-Mutation	OK	mtime weiterhin `1777991334`
Kein G10 / Worker / Mainnet / Push	OK	nichts angefasst

---

7. Empfehlung für B-FEE-FIX-4

Phase 1 (Quick-Fix, niedriges Risiko):

- Helper-Rule erweitern: `if fee_cost == 0: fallback notional × TRADE_FEE`
- Damit decken Testnet-Buys + Mainnet-fee-fehlt-Fälle ab
- 1 Zeile Code-Change, gleiche Helper-Function, ~5 neue Tests

Phase 2 (Architektur-Verbesserung, mittleres Risiko):

- `fetchMyTrades`-basierte Fee-Resolution mit BNB-discount-Support
- multi-fill-Aggregation
- fee_source-Logging in closed_trades

Empfehlung: mit Phase 1 starten (kleiner Fix, sofortiger Wert), Phase 2 als separate Phase mit eigenem Preflight.

Stand: 2026-05-08 UTC, vor Implementierung. Erfordert explizites GO für Phase 1.